Diffeomorphic interpolation for efficient persistence-based topological optimization

Mathieu Carriere, Marc Theveneau, Théo Lacombe

Diffeomorphic interpolation for efficient persistence-based topological optimization

Mathieu Carrière
DataShape
Centre Inria d'Université Côte d'Azur
Sophia Antipolis, France
mathieu.carriere@inria.fr

Marc Theveneau*
Shape Analysis Group
Computer Science department, McGill
Montréal, Quebec, Canada
marc.theveneau@mila.quebec

Théo Lacombe
Laboratoire d'Informatique Gaspard Monge,
Univ. Gustave Eiffel, CNRS, LIGM, F-77454
Marne-la-Vallée, France
theo.lacombe@univ-eiffel.fr

Abstract

Topological Data Analysis (TDA) provides a pipeline to extract quantitative topological descriptors from structured objects. This enables the definition of topological loss functions, which assert to what extent a given object exhibits some topological properties. These losses can then be used to perform topological optimization via gradient descent routines. While theoretically sounded, topological optimization faces an important challenge: gradients tend to be extremely sparse, in the sense that the loss function typically depends on only very few coordinates of the input object, yielding dramatically slow optimization schemes in practice.

Focusing on the central case of topological optimization for point clouds, we propose in this work to overcome this limitation using *diffeomorphic interpolation*, turning sparse gradients into smooth vector fields defined on the whole space, with quantifiable Lipschitz constants. In particular, we show that our approach combines efficiently with subsampling techniques routinely used in TDA, as the diffeomorphism derived from the gradient computed on a subsample can be used to update the coordinates of the full input object, allowing us to perform topological optimization on point clouds at an unprecedented scale. Finally, we also showcase the relevance of our approach for black-box autoencoder (AE) regularization, where we aim at enforcing topological priors on the latent spaces associated to fixed, pre-trained, black-box AE models, and where we show that learning a diffeomorphic flow can be done once and then re-applied to new data in linear time (while vanilla topological optimization has to be re-run from scratch). Moreover, reverting the flow allows us to generate data by sampling the topologically-optimized latent space directly, yielding better interpretability of the model.

1 Introduction

Persistent homology (PH) is a central tool of Topological Data Analysis (TDA) that enables the extraction of quantitative topological information (such as, e.g., the number and sizes of loops, connected components, branches, cavities, etc) about structured objects (such as graphs, times series or

*Part of this work was done when MT was intern in the Laboratoire d'Informatique Gaspard Monge and student in Université Paris-Saclay.

Preprint. Under review.

point clouds sampled from, e.g., submanifolds), summarized in compact descriptors called *persistence diagrams* (PDs). PDs were initially used as features in Machine Learning (ML) pipelines; due to their strong invariance and stability properties, they have been proved to be powerful descriptors in the context of classification of time series [39, 19], graphs [5, 23, 24], images [2, 25, 16], shape registration [7, 6, 34], or analysis of neural networks [22, 3, 26], to name a few.

Another active line or research at the crossroad of TDA and ML is *(persistence-based) topological optimization*, where one wants to modify an object X so that it satisfies some topological constraints as reflected in its persistence diagram $\mathrm{Dgm}(X)$. The first occurrence of this idea appears in [21], where one wants to deform a point cloud $X \in \mathbb{R}^{n\times d}$ so that $\mathrm{Dgm}(X)$ becomes as close as possible (w.r.t. an appropriate metric denoted by W) to some target diagram D_{target}, hence yielding to the problem of minimizing $X \mapsto W(\mathrm{Dgm}(X), D_{\mathrm{target}})$. This idea has then been revisited with different flavors, for instance by adding topology-based terms in standard losses in order to regularize ML models [14, 31, 29], improving ML model reconstructions by explicitly accounting for topological features [16], or improving correspondences between 3D shapes by forcing matched regions to have similar topology [34]. Formally, this goes through the minimization of an objective function

$$L : X \mapsto \ell(\mathrm{Dgm}(X)) \in \mathbb{R}, \tag{1}$$

where ℓ is a user-chosen loss function that quantifies to what extent $\mathrm{Dgm}(X)$ reflects some prescribed topological properties inferred from X. Under mild assumptions (see Section 2), the map L is differentiable generically and its gradients are obtained as a byproduct of the computation of $\mathrm{Dgm}(X)$. However, these approaches are limited in practice by two major issues: (i) the computation of $X \mapsto \mathrm{Dgm}(X)$ scales poorly with the size of X (e.g., number of points n in a point cloud, number of nodes in a graph, etc), and (ii) the gradient $\nabla L(X)$ tends to be very *sparse*: if $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n\times d}$ is a point cloud, $\nabla L(X)_i \neq 0$ for only very few indices $i \in \{1, \ldots, n\}$ (the corresponding points are called the *critical points* of the topological gradient, see Section 2.1).

Related works. Several articles have studied topological optimization in the TDA literature. The standard, or *vanilla*, framework to define and study gradients obtained from topological losses was described in [4, 28], where the high sparsity and long computation times were first identified. To mitigate this issue, the authors of [32] introduces the notion of *critical set* that extends the usually sparse set of critical points in order to get a gradient-like object that would affect more points in X. In [36], the authors use an average of the vanilla topological gradients of several subsamples to get a denser and faster gradient. On the theoretical side, the authors of [27] demonstrated that adapting the stratified structure induced by PDs to the gradient definition enables faster convergence.

Limitations. Despite proposing interesting ways to accelerate gradient descent, the approaches mentioned above are still limited in the sense that their proposed gradients are not defined on the whole space, but only on a sparse subset of the current observation X, which still prevents their use in different contexts, that we investigate in our experiments (Section 4). First, when the data has more than tens of thousands of points, the number of subsamples needed to capture relevant topological structures (when using [36]), as well as the critical set computations (when using [32]), both become *practically infeasible*. Second, when optimizing the topology of datasets obtained as *latent spaces* of a *black-box autoencoder model* (i.e., an autoencoder with forbidden access to its architecture, parameters, and training), then (a) the topological gradients of [36, 32] *cannot* be reused to process new such datasets, and topological optimization has to be performed from scratch every time that new data comes in, (b) this also impedes their *transferability*, as re-running gradient descent every time makes it very difficult to guarantee some stability for the final output, and finally (c) one *cannot* generate new data by sampling the optimized latent spaces directly, as it would require to apply the sequence of reverted gradients (which are not well-defined everywhere).

Contributions and Outline. In this article, we propose to replace the standard gradient $\nabla L(X)$ of (1) derived formally by a *diffeomorphism* $v : \mathbb{R}^d \to \mathbb{R}^d$ that *interpolates* $\nabla L(X)$ on its non-zero entries, that is $v(x_i) = \nabla L(X)_i$ for $i \in I := \{j \,|\, \nabla L(X)_j \neq 0\}$ and is, in some sense, as smooth as possible. More precisely, our contribution is three-fold:

- We introduce a *diffeomorphic interpolation* of the vanilla topological gradient, which extends this gradient to a smooth and denser vector field defined on the whole space $\mathbb{R}^d$, and which is able to move a lot more points in X at each iteration,

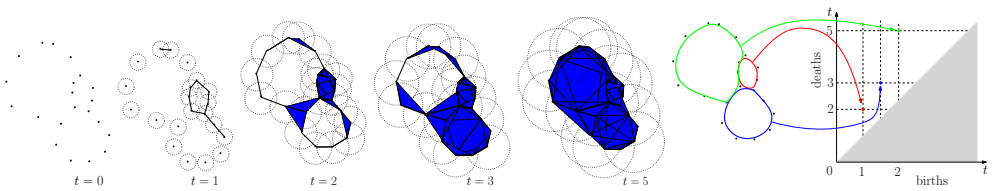


Figure 1: Illustration of the Vietoris-Rips filtration on a point cloud in $\mathbb{R}^d$, focusing on one-dimensional topological features (loops). When the filtration parameter t increases, loops appear and disappear in the filtration. These values are accounted in the resulting persistence diagram (right).

- We prove that its updates indeed decrease topological losses, and we quantify its smoothness by upper bounding its Lipschitz constant (in the context of topological losses),
- We showcase its practical efficiency: we show that it compares favorably to the main baseline [32] in terms of convergence speed, that its combination with subsampling [36] allows to process datasets whose sizes are currently out of reach in TDA, and that it can successfully be used for the tasks mentioned above concerning black-box autoencoder models.

Section 2 provides necessary background in Topological Data Analysis and diffeomorphic interpolations. Section 3 presents our approach and its corresponding guarantees, and Section 4 showcases our experiments. Limitations and further research directions are discussed in Section 5.

2 Background

2.1 Topological Data Analysis

In this section, we recall the basic materials of Topological Data Analysis (TDA), and refer the interested reader to [20, 33] for a thorough overview. We restrict the presentation to our case of interest: extracting topological information from a point cloud using the standard *Vietoris-Rips* (VR) filtration. A more extensive presentation of the TDA machinery is provided in Appendix A.

Let $X = (x_1, \dots, x_n) \in \mathbb{R}^{n \times d}$. The Vietoris-Rips filtration consists of building an increasing sequence of simplicial complexes $(K_t)_t$ over X by inserting a simplex $\sigma = (x_{i_1}, \dots, x_{i_p})$ whenever $\forall j, j' \in \{1, \dots, p\}$, $\|x_{i_j} - x_{i_{j'}}\| \leq t$. Each time a simplex σ is inserted, it either creates a topological feature (e.g., inserting a face creates a cavity, that is a 2-dimensional topological feature) or destroy a pre-existing feature (e.g., the face insertion fills a loop, that is a 1-dimensional feature, making it topologically trivial). The persistent homology machinary tracks the apparition and destruction of such features in the so-called *persistence diagram* (PD) of X, denoted by $\mathrm{Dgm}(X)$. Therefore, $\mathrm{Dgm}(X)$ is a set of points in $\mathbb{R}^2$ of the form (t_b, t_d) with $t_d \geq t_b$, where each such point accounts for the presence of a topological feature inferred from X that appeared at time t_b following the insertion of an edge (x_{i_1}, x_{i_2}) with $\|x_{i_1} - x_{i_2}\| = t_b$ and disappeared at time t_d following the insertion of an edge (x_{i_3}, x_{i_4}) with $\|x_{i_3} - x_{i_4}\| = t_d$. Figure 1 illustrates this construction. From a computational standpoint, computing the VR diagram of $X \in \mathbb{R}^{n \times d}$ that would reflect topological features of dimension $d' \leq d$ runs in $O(n^{d'+2})$, making the computation quickly unpractical when n increases, even when restricting to low-dimensional features as loops ($d' = 1$) or cavities ($d' = 2$).

Topological optimization. PDs are made to be used in downstream pipelines, either as static features (e.g., for classification purposes) or as intermediate representations of X in optimization schemes. In this work, we focus on the second problem. We formally consider the minimization of objective functions of the form

$$L : X \in \mathbb{R}^{n \times d} \mapsto \ell(\mathrm{Dgm}(X)) \in \mathbb{R}. \tag{2}$$

Here, ℓ represents a loss function taking value from the space of persistence diagrams, denoted by $\mathcal{D}$ in the following. The space $\mathcal{D}$ can be equipped with a canonical metric, denoted by W and whose formal definition is not required in this work, for which a central result is that the map $X \mapsto \mathrm{Dgm}(X)$ is stable (Lipschitz continuous) [17, 18, 35]. Therefore, if ℓ is Lipschitz continuous, so is L hence it admits a gradient almost everywhere by Rademacher theorem. Building on these theoretical statements, one can consider the "vanilla" gradient descent update $X_{k+1} \coloneqq X_k - \lambda \nabla L(X_k)$

for a given learning-rate $\lambda > 0$ and iterate it in order to minimize (2). Theoretical properties of this seminal scheme (and natural extensions, e.g., stochastic) have been studied in [4, 27], where convergence (to a local minimum of L) is proved.

From a computational perspective, deriving $\nabla L(X)$ comes in two steps. Let $\mu := \mathrm{Dgm}(X)$, written as $\mu = \{(b_i, d_i) \,|\, i \in I\}$ for some finite set of indices I. To each $i \in I$ correspond four (possibly coinciding) points $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}$ in the input point cloud X. Intuitively, minimizing $\mu \mapsto \ell(\mu)$ boils down to prescribe a descent direction $(\delta b_i, \delta d_i) \in \mathbb{R}^2$ to each (b_i, d_i) for $i \in I$, where $\delta b_i = \frac{\partial \ell}{\partial b_i}(\mu)$ and $\delta d_i = \frac{\partial \ell}{\partial d_i}(\mu)$. Backpropagating this perturbation to X will move the corresponding points $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}$ in order to increase or decrease the distances $\|x_{i_1} - x_{i_2}\| = b_i$ and $\|x_{i_3} - x_{i_4}\| = d_i$ accordingly. This yields to the following formula:

$$\frac{\partial L}{\partial x}(X) = \sum_{i,\ x \to (b_i, d_i)} \left[\frac{\partial \ell}{\partial b_i} \cdot \frac{\partial b_i}{\partial x} + \frac{\partial \ell}{\partial d_i} \cdot \frac{\partial d_i}{\partial x} \right](X), \tag{3}$$

where the notation $x \to (b_i, d_i)$ means that $x \in X$ appears in (at least) one of the four points yielding the presence of (b_i, d_i) in the diagram $\mu = \mathrm{Dgm}(X)$. A fundamental contribution of [28, §3.3] is to prove that the chain rule formula (3) is indeed valid[2]. Most of the time, a point $x \in X$ will not belong to any critical pair (σ_b, σ_d) and the above gradient coordinate is 0. Therefore, the gradient of L depends on very few points of X, yielding the sparsity phenomenon discussed in Section 1.

Examples of common topological losses. Let $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ be a point cloud and $\mathrm{Dgm}(X) = \{(b_i, d_i) \,|\, i \in I\}$ be its persistence diagram. There are several natural losses that have been introduced in the TDA literature:

- Topological simplification losses: typically of the form $\sum_{i \in \tilde{I}} (b_i - d_i)^2$, where $\tilde{I} \subset I$. Such losses push (some of the) points in $\mathrm{Dgm}(X)$ toward the diagonal $\Delta = \{b = d\}$, hence destroying the corresponding topological features appearing in X.
- Topological augmentation losses [4]: similar to simplification losses, but typically attempting to push points in $\mathrm{Dgm}(X)$ away from Δ, i.e., minimizing $-\sum_{i \in \tilde{I}} (b_i - d_i)^2$, to make topological features of X more salient. As such losses are not coercive, they are usually coupled with regularization terms to prevent points in X going to infinity.
- Topological registration losses [21]: given a target diagram D_{target}, one minimizes $W(\mathrm{Dgm}(X), D_{\mathrm{target}})$ where W denotes a standard metric between persistence diagrams. This loss attempts to modify X so that it exhibits a prescribed topological structure.

2.2 Diffeomorphic interpolations

In order to overcome the sparsity of gradients appearing in TDA, we rely on diffeomorphic interpolations (see, e.g., [40, Chapter 8]). Let $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$, let $I \subseteq \{1, \ldots, n\}$ denote the set of indices on which $\nabla L(X)$ is non-zero and let $a_i := (\nabla L(X))_i \in \mathbb{R}^d$ for $i \in I$. Our goal is to find a smooth vector field $\tilde{v} : \mathbb{R}^d \to \mathbb{R}^d$ such that, for all $i \in I$, $\tilde{v}(x_i) = a_i$. To formalize this, we consider a Hilbert space $H \subset (\mathbb{R}^d)^{\mathbb{R}^d}$ for which the map $\delta_x^\alpha : f \mapsto \langle \alpha, f(x) \rangle_{\mathbb{R}^d} = \alpha^T f(x)$ is continuous for any $(\alpha, x) \in \mathbb{R}^d \times \mathbb{R}^d$. Such a space is called a Reproducing Kernel Hilbert Space (RKHS)[3]. A crucial property is that there exists a matrix-valued kernel operator $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d \times d}$ whose outputs are symmetric and positive definite, and related to H through the relation $\langle k_x^\alpha, k_y^\beta \rangle_H = \alpha^T K(x, y) \beta$ for all $x, y, \alpha, \beta \in \mathbb{R}^d$, where $k_x^\alpha \in H$ is the unique vector provided by the Riesz representation theorem such that $\langle k_x^\alpha, f \rangle = \langle \alpha, f(x) \rangle$. Conversely, any such kernel K induces a (unique) RKHS H (of which K is the reproducing kernel). Now, we can consider the following problem:

$$\text{minimize } \|v\|_H, \text{ s.t. } v(x_i) = a_i, \ \forall i \in I,$$

that is, we are seeking for the smoothest (lowest norm) element of H that solves our interpolation problem. The solution $\tilde{v}$ of this problem is the projection of 0 onto the affine set $\{v \in H \,|\, v(x_i) = a_i, \forall i \in I\}$. Observe that $\tilde{v}$ belongs to the orthogonal of $\{v \in H \,|\, v(x_i) = 0, \forall i \in I\}$, and thus of

[2] This is not trivial, because the intermediate space $\mathcal{D}$ is only a metric space.

[3] In many applications, RKHS are restricted to spaces of functions valued in $\mathbb{R}$ or $\mathbb{C}$, but the theory adapts faithfully to the more general setting of vector-valued maps.

$\{v \in H \mid \langle k_{x_i}^{\alpha_i}, v\rangle_H = 0,\ \forall i \in I, \alpha_i \in \mathbb{R}^d\}$, and therefore $\tilde{v} \in \mathrm{span}(\{k_{x_i}^{\alpha_i} \mid i \in I\})$. This justifies to search for $\tilde{v}$ in the form of $\tilde{v}(x) = \sum_{i\in I} K(x, x_i)\alpha_i$, and the interpolation that it must satisfy yields $\tilde{v}(x) = \sum_{i\in I} K(x, x_i)(\mathbb{K}^{-1}a)_i$, where $\mathbb{K}$ is the block matrix $(K(x_i, x_j))_{i,j\in I}$ and $a = (a_i)_{i\in I}$. See also [40, Theorem 8.8]. In particular, it is important to note that $\tilde{v}$ inherits from the regularity of K and will typically be a diffeomorphism in this work. If K is the Gaussian kernel defined by $K(x, y) \coloneqq \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) I_d$ for some bandwidth $\sigma > 0$, a choice to which we stick to in the rest of this work, the expression of $\tilde{v}$ reduces to

$$\tilde{v}(x) = \sum_{i\in I} \rho_\sigma(\|x - x_i\|)\alpha_i, \tag{4}$$

where $\rho_\sigma(u) \coloneqq e^{-\frac{u^2}{2\sigma^2}}$, and $\alpha_i \coloneqq (\mathbf{K}^{-1}a)_i$ with $\mathbf{K} = (\rho_\sigma(\|x_i - x_j\|)I_d)_{i,j\in I}$. Note that $\tilde{v}$ can be understood as the convolution of a with a Gaussian kernel, but involving a correction $\mathbf{K}^{-1}$ guaranteeing that after the convolution, the interpolation constraint is satisfied. We will call $\tilde{v}$ the *diffeomorphic interpolation* associated to the vectors and indices $\{a_i \mid i \in I\}$.

3 Diffeomorphic interpolation of the vanilla topological gradient

3.1 Methodology

We aim at minimizing a loss function $L : X \mapsto \ell(\mathrm{Dgm}(X))$ as in (2), starting from some initialization X_0, and assuming that L is lower bounded (typically by 0) and locally semi-convex. This assumption is typically satisfied by the topological losses ℓ introduced in Section 2.1. Gradient descents implemented in practice are (explicit) discretization of the *gradient flow*

$$\frac{\mathrm{d}X}{\mathrm{d}t} \in -\partial L(X(t)), \quad X(0) = X_0, \tag{5}$$

where $\partial L(X) \coloneqq \{v \mid L(Y) \geq L(X) + v \cdot (Y - X) + o(Y - X)$ for all $X, Y\}$ denotes the subdifferential of L at X. Note that a topological loss L is typically *not* differentiable everywhere, since the map $X \mapsto \mathrm{Dgm}(X)$ is differentiable almost everywhere but not in $C^{1,1}$. However, uniqueness of the gradient flow on a maximal interval $[0, +\infty[$ is guaranteed if L is lower bounded and locally semi-convex [15, §B.1].

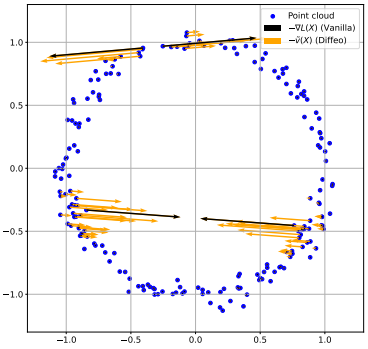


Figure 2: (blue) A point cloud X, and (black) the negative gradient $-\nabla L(X)$ of a simplification loss which aims at destroying the loop by collapsing the circle (reduce the loop's death time) and tearing it (increase the birth time). While $\nabla L(X)$ only affects four points in X, the diffeomorphic interpolation $\tilde{v}(X)$ (orange, $\sigma = 0.1$) is defined on $\mathbb{R}^d$, hence extends smoothly to other points in X.

In this work, we propose to use the dynamic described by the diffeomorphism $\tilde{v}_t$ introduced in (4) interpolating the current vanilla topological gradient $\nabla L(X_t)$ at each time t, formally considering solutions $\tilde{X}$ of

$$\frac{\mathrm{d}\tilde{X}}{\mathrm{d}t} = -\tilde{v}_t(\tilde{X}(t)), \quad \tilde{X}(0) = X_0. \tag{6}$$

Here, slightly overloading notation, $\tilde{v}_t(\tilde{X}(t))$ denotes the $n \times d$ matrix where the i-th line is given by $\tilde{v}_t(\tilde{X}(t)_i)$. The *flow* at time T associated to (6) is the map

$$\varphi_T : x_0 \mapsto x_0 - \int_0^T \tilde{v}_t(x(t))\mathrm{d}t, \quad \dot{x}(t) = -\tilde{v}_t(x(t)),\ x(0) = x_0, \tag{7}$$

which can inverted by simply following the flow backward (i.e., by following $\tilde{v}_t$ instead of $-\tilde{v}_t$). We now guarantee that at each time t, following $\tilde{v}_t$ instead of the vanilla topological gradient $\nabla L(X_t)$ still provides a descent direction for the topological loss L.

Proposition 3.1. *For each $t \geq 0$, it holds that $\frac{\mathrm{d}L(\tilde{X}(t))}{\mathrm{d}t} = -\|\nabla L(\tilde{X}(t))\|^2 \leq 0$.*

Proof. One has $\frac{\mathrm{d}L(\tilde{X}(t))}{\mathrm{d}t} = -\langle \nabla L(\tilde{X}(t)), \tilde{v}_t(\tilde{X}(t))\rangle = -\sum_{i=1}^n (\nabla L(\tilde{X}(t)))_i \cdot (\tilde{v}_t(\tilde{X}(t)))_i$. Since $\nabla L(\tilde{X}(t))_i = 0$ for $i \notin I$, and $\tilde{v}_t(\tilde{X}(t))_i = -\nabla L(\tilde{X}(t))_i$ for $i \in I$, the result follows. □

Moreover, it is also possible to upper bound the smoothness, i.e., the Lipschitz constant, of $\tilde{v}$:

Proposition 3.2. *Let L be the simplification or augmentation loss computed with $k = |\tilde{I}|$ PD points, as defined at the end of Section 2.1. Let $\tilde{v} = \tilde{v}_t$ be the diffeomorphic interpolation associated to the vanilla topological gradient at time $t \geq 0$. Then, one has, $\forall x, y \in \mathbb{R}^d$ and $t \geq 0$:*

$$\|\tilde{v}(x) - \tilde{v}(y)\|_2 \leq \|\tilde{v}(x) - \tilde{v}(y)\|_1 \leq C_d \cdot \sigma^{d-1} \cdot \kappa(\mathbf{K}) \cdot \mathrm{Pers}_k(\mathrm{Dgm}(\tilde{X}(t))) \cdot \|x - y\|_2,$$

where $C_d = \sqrt{d} \cdot 2^{3+\frac{d+1}{2}} \cdot \pi^{\frac{d-1}{2}}$, $\kappa(\mathbf{K})$ is the condition number of $\mathbf{K}$, and $\mathrm{Pers}_k(\mathrm{Dgm}(\tilde{X}(t)))$ is the sum of the k largest distances to the diagonal in $\mathrm{Dgm}(\tilde{X}(t))$.

The proof is deferred to Appendix B. This upper bound can be used to quantify how smooth the diffeomorphic interpolation is (when computed from topological losses) as close points can still have significantly different gradients if the Lipschitz constant is large. Overall, smoothness is affected by the dimension d (curse of dimensionality), the bandwidth of the Gaussian kernel σ (as kernels with large σ can affect more points), the condition number $\kappa(\mathbf{K})$ (ill-conditioned kernel matrices are more sensitive), and the distances to the diagonal in the current PD (larger topological features lead to larger vanilla topological gradients for simplification or augmentation losses).

3.2 Subsampling techniques to scale topological optimization

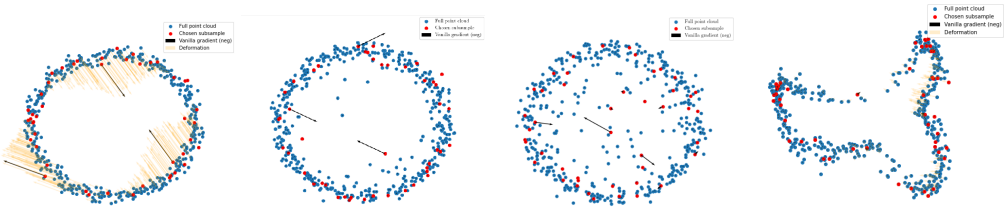
As a consequence of the limited scaling of the Vietoris-Rips filtration with respect to the number of points n of the input point cloud X, it often happens in practical applications that computing the VR diagram $\mathrm{Dgm}(X)$ of a large point set X (*a fortiori* its gradient) turns out to be intractable. A natural workaround is to randomly sample s-points from X, with $s \ll n$, yielding a smaller point cloud $X' \subset X$. Provided that the Hausdorff distance between X' and X is small, the stability theorem [18, 17, 8] ensures that $\mathrm{Dgm}(X')$ is close to $\mathrm{Dgm}(X)$. See [11, 12, 9] for an overview of subsampling methods in TDA.

However, the sparsity of vanilla topological gradients computed from topological losses strikes further when relying on subsampling: only a tiny fraction of the seminal point cloud X is likely to be updated at each gradient step. In contrast, using the diffeomorphic interpolation $\tilde{v}$ (of the vanilla topological gradient) computed on the subsample X' still provides a vector field defined on the whole input space $\mathbb{R}^d$, in particular on each point of X and the update can then be performed in linear time with respect to n. This yields Algorithm 1. Figure 3 illustrates the qualitative benefits offered by the joint use of subsampling and diffeomorphic interpolations when compared to vanilla topological gradients. A larger-scale experiment is provided in Section 4.

Algorithm 1 Diffeomorphic gradient descent for topological loss functions with subsampling

Input: Initial $X_0 \in \mathbb{R}^{n \times d}$, loss function ℓ, learning rate $\lambda > 0$, subsampling size $s \in \{1, \ldots, n\}$, max. epoch $T \geq 1$, stopping criterion.
Set $L : X \mapsto \ell(\mathrm{Dgm}(X))$ (+ possibly a regularization term in X).
for $k = 1, \ldots, T$ **do**
  Subsample $X'_{k-1} = \{x'_1, \ldots, x'_s\}$ uniformly from X_{k-1}.
  Compute $\nabla L(X'_{k-1})$ (vanilla topological gradient)
  Compute the diffeomorphic interpolation $\tilde{v}(X'_{k-1})$ from $\nabla L(X'_{k-1})$ using (4).
  Set $X_t \coloneqq X_{k-1} - \lambda \tilde{v}(X_{k-1})$.
  if stopping criterion is reached **then**
    Return X_k
  end if
end for
Return X_T

Stopping criterion. A natural stopping criterion for Algorithm 1 is to assess whether the loss $L(X_t) = \ell(\mathrm{Dgm}(X_t))$ is smaller than some $\varepsilon > 0$. However, computing $\mathrm{Dgm}(X_t)$ can be intractable if X_t is large. Therefore, a tractable loss to consider is $\hat{L}(X_t) \coloneqq \mathbb{E}[\ell(\mathrm{Dgm}(X'_t))]$, where X'_t is a uniform s-sample from X_t. Under that perspective, Algorithm 1 can be re-interpreted as a kind of stochastic gradient descent on $\hat{L}$, for which two standard stopping criteria can be



(a) $t = 0$ (b) $t = 100$ (Vanilla) (c) $t = 500$ (Vanilla) (d) $t = 100$ (Diffeo)

Figure 3: Showcase of the usefulness of subsampling combined with diffeomorphic interpolations to minimize a topological simplification loss, with parameters $\lambda = 0.1$, $s = 50$, $n = 500$. (a) Initial point cloud X (blue), subsample X' (red), vanilla topological gradient on the subsample (black) and corresponding diffeomorphic interpolation (orange). (b) and (c), the point cloud X_t after running $t = 100$ and $t = 500$ steps of vanilla gradient descent. (d) the point cloud X_t after running $t = 100$ steps of diffeomorphic gradient descent.

used: (a) compute an exponential moving average of the loss on individual samples X'_t over iterations, or (b) compute a validation loss, i.e., sample $X'_{t,(1)}, \dots, X'_{t,(K)}$ and estimate $\hat{L}$ by $K^{-1} \sum_{k=1}^{K} \ell(\mathrm{Dgm}(X'_{t,(k)}))$. Empirically, we observe that the latter approach with $K = n/s$ (more repetitions for smaller sample sizes to mitigate variance) yields the most satisfactory results (faster convergence toward a better objective X_t) overall, and thus stick to this choice in our experiments.

4 Numerical experiments

We provide numerical evidence for the strength of our diffeomorphic interpolations. PH-related computations relies on the library `Gudhi` [37] and automatic differentiation relies on `tensorflow` [1]. The "big-step gradient" baseline [32] implementation is based on `oineus`[4]. The first two experiments were run on a `11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz`, the last one on a `2x Xeon SP Gold 5115 @ 2.40GHz`.

Convergence speed and running times. We sample uniformly $N = 200$ points on a unit circle in $\mathbb{R}^2$ with some additional Gaussian noise, and then minimize the simplification loss $L : X \mapsto \sum_{(b,d)\in \mathrm{Dgm}(X)} |d|^2$, which attempts to destroy the underlying topology in X by reducing the death times of the loops by collapsing the points. The respective gradient descents are iterated over a maximum of 250 epochs, possibly interrupted before if a loss of 0 is reached ($\mathrm{Dgm}(X)$ is empty), with a same learning rate $\lambda = 0.1$. The bandwidth of the Gaussian kernel in (4) is set to $\sigma = 0.1$. We include the competitor `oineus` [32], as—even though relying on a fairly different construction—this method shares a key idea with ours: extending the vanilla gradient to move more points in X. We stress that both approaches can be used in complementarity: compute first the "big-step gradient" of [32] using `oineus`, and then extend it by diffeomorphic interpolation. Results are displayed in Figure 4. In terms of loss decrease over *iterations*, both "big-step gradients" and our diffeomorphic interpolations significantly outperform vanilla topological gradients, and their combined use yields the fastest convergence (by a slight margin over our diffeomorphic interpolations alone). However, in terms of raw running times, the use of `oineus` involves a significant computational overhead, making our approach the fastest to reach convergence by a significant margin.

Subsampling. We now showcase how using our diffeomorphic interpolation jointly with subsampling routines (Algorithm 1) allows to perform topological optimization on point clouds with thousands of points, a new scale in the field. For this, we consider the vertices of the Stanford Bunny [38], yielding a point cloud $X_0 \in \mathbb{R}^{n\times d}$ with $n = 35,947$ and $d = 3$. We consider a topological augmentation loss (see Section 2.1) for two-dimensional topological features, i.e., we aim at increasing the persistence of the bunny's cavity. The size of n makes the computation of $\mathrm{Dgm}(X_0)$ untractable (recall that it scales in $O(n^4)$); we thus rely on subsampling with sample size $s = 100$ and compare the vanilla gradient descent scheme and our scheme described in Algorithm 1. Results are displayed in Figure 5. Because it only updates a tiny fraction of the initial point cloud at each

[4] https://github.com/anigmetov/oineus

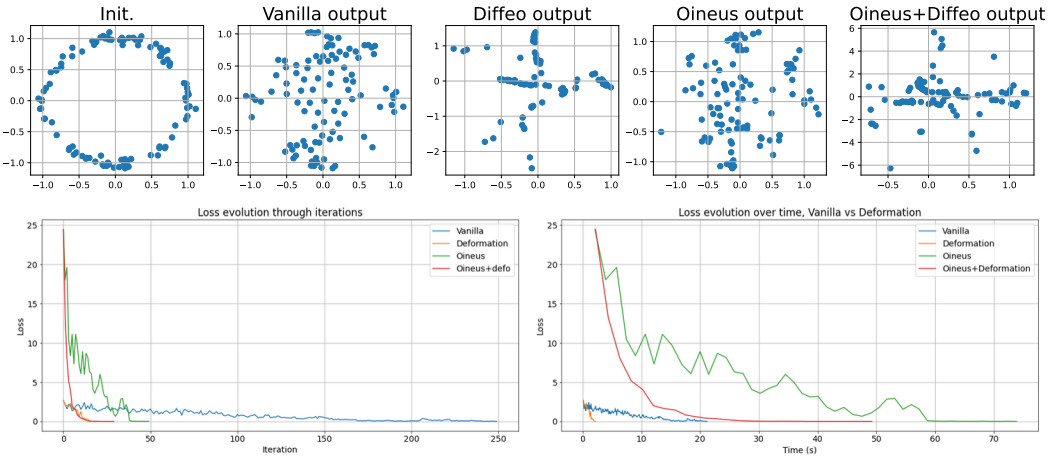


Figure 4: (Top) From left to right: initial point cloud, and final point cloud for the different flows. (Bottom) Evolution of the loss with respect to the number of iterations and with respect to running time.

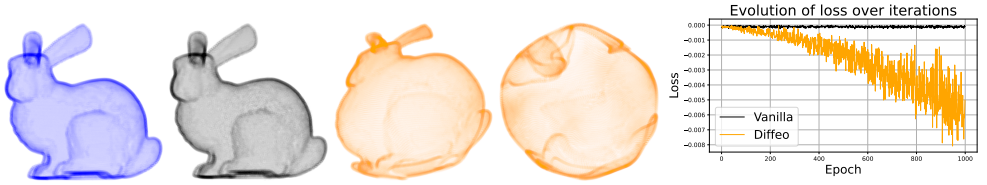


Figure 5: From left to right: initial Stanford bunny X_0, the point cloud after $1,000$ epochs of vanilla topological gradient descent (barely any changes), the point cloud after 200 epochs of diffeomorphic gradient descent, after 1,000 epochs, and eventually the evolution of losses for both methods over iterations.

iteration, the vanilla topological gradient with subsampling barely changes the point cloud (nor decreases the loss) in 1,000 epochs. In sharp contrast, as our diffeomorphic interpolation computed on subsamples is defined on $\mathbb{R}^3$, it updates the whole point cloud at each iteration, making possible to decrease the objective function where the vanilla gradient descent is completely stuck. Note that a step of diffeomorphic interpolation, in that case, takes about 10 times longer than a vanilla step. An additional subsampling experiment can be found in Appendix C.

Black-box autoencoder models. Finally, we apply our diffeomorphic interpolations to black-box autoencoder models. In their simplest formulation, autoencoders (AE) can be summarized as two maps $E : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $D : \mathbb{R}^{d'} \to \mathbb{R}^d$ called encoder and decoder respectively. The intermediate space $\mathbb{R}^{d'}$ in which the encoder E is valued is referred to as a *latent space* (LS), with typically $d' \ll d$. In general, without further care, there is no reason to expect that the LS of a point cloud X, $E(X) = \{E(x_1), \dots, E(x_n)\}$, reflects any geometric or topological properties of X. While this can be mitigated by adding a topological regularization term to the loss function *during the training* of the autoencoder [29, 4], this *cannot* work in the setting where one is given a *black-box, pre-trained* AE. However, replacing (E, D) by $(\varphi \circ E, D \circ \varphi^{-1})$ for any invertible map $\varphi : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$ yields an AE producing the same outputs yet changing the LS $E(X)$, without explicit access to the AE's model. Hence, we propose to learn such a φ with diffeomorphic interpolations: given some latent space $E(X)$, we apply T steps of our diffeomorphic gradient descent algorithm to $X \mapsto \ell(\mathrm{Dgm}(X))$ initialized at $E(X)$. We thus get a sequence of smooth displacements $-\tilde{v}_1, \dots, -\tilde{v}_T$ of $\mathbb{R}^{d'}$ that discretizes the flow (7) via $\varphi : x_0 \mapsto x_0 - \sum_{k=1}^{T} \tilde{v}_k(x_{k-1})$ where $x_k - x_{k-1} = -\tilde{v}_k(x_{k-1})$, and such that $\mathrm{Dgm}(\varphi(E(X)))$ is more topologically satisfying. This fixed diffeomorphism φ can then be re-applied to any new data coming out of the encoder in a deterministic way. Moreover, any random sample from the topologically-optimized LS can be inverted without further computations by following $\tilde{v}_T, \tilde{v}_{T-1}, \dots, \tilde{v}_1$, which allows to push the new sample back to the initial LS, and then apply the decoder on it. Again, this cannot be achieved with baselines [36, 32].

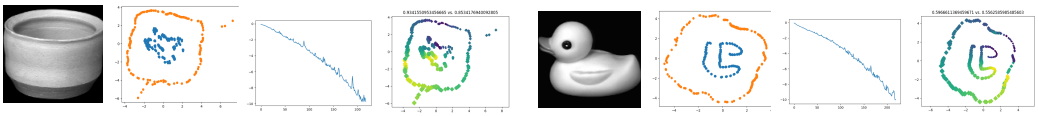


Figure 7: `COIL` images, their corresponding initial LS in blue and final LS obtained with diffeomorphic gradient descent in orange, the topological loss, and the same LSs colored with ground truth angles with final (left) vs. initial (right) correlations with predicted angles displayed as titles, for both vase (left) and duck (right).

In order to illustrate these properties, we trained a variational autoencoder (VAE) to project a family of datasets of images representing rotating objects, named `COIL` [30], to two-dimensional latent spaces. Given that every dataset in this family is comprised of 288 pictures of the same object taken with different angles, one can impose a prior on the topology of the corresponding LSs, namely that they are sampled from circles. However, the VAE architecture is shallow (the encoder has one fully-connected layer (100 neurons), and the decoder has two (50 and 100 neurons), all layers use ReLu activations), and thus the learned latent spaces, although still looking like curves thanks to continuity, do not necessarily display circular patterns. This makes generating new data more difficult, as latent spaces are harder to interpret. To improve on this, we learn a flow φ as described above with an augmentation loss associated to the 1-dimensional PD point which is the most far away from the diagonal, in order to force latent spaces to have significant one-dimensional Vietoris-Rips persistent homology. As the datasets are small, we do not use subsampling, and we use learning rate $\lambda = 0.1$, Gaussian kernels with bandwidth $\sigma = 0.3$ and an increase of at least 3. in the topological loss (from an iteration to the next) to stop the algorithm[5].

We provide some qualitative results in Figure 7 (see also Appendix C, Figure 10). In order to quantify the improvement, we also computed the correlation between the ground-truth angles θ_i and the angles $\hat{\theta}_i$ computed from the topologically-optimized LS embeddings with

$$\hat{\theta}_i := \angle(\varphi \circ E(x_i) - \hat{\mathbb{E}}[\varphi \circ E(X)], \varphi \circ E(x_1) - \hat{\mathbb{E}}[\varphi \circ E(X)]),$$

where $\hat{\mathbb{E}}[\varphi \circ E(X)] := n^{-1} \sum_{i=1}^{n} \varphi \circ E(x_i)$, and φ denotes our flow or the sequence of vanilla gradients. See the table below.

Optim.	Duck	Cat	Pig	Vase	Teapot
Vanilla	0.56	0.79	0.17	0.85	0.32
Diffeo	**0.6**	**0.83**	**0.74**	**0.93**	**0.39**

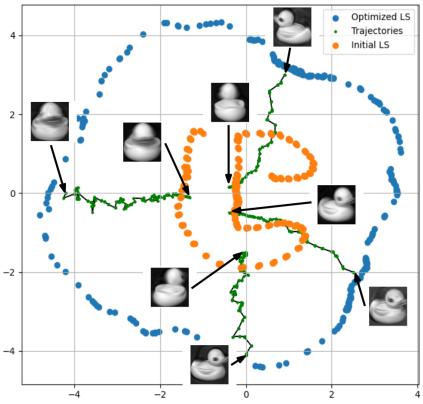


Figure 6: As the four samples from the topologically-optimized LS (blue) are far from the initial LS (orange), the decoded images are fuzzy. However, reverting φ and following the corresponding green trajectories allows to render good-looking images.

As expected, correlation becomes better after forcing the latent spaces to have the topology of a circle. This better interpretability is also illustrated in Figure 6, in which four angles are specified, which are mapped to the topologically-optimized LS, then pushed to the initial LS of the black-box VAE by following the reverted flow of our learned diffeomorphism φ, and finally decoded back into images.

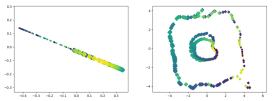
5 Conclusion

In this article, we have presented a way to turn sparse topological gradients into dense diffeomorphisms with quantifiable Lipschitz constants, and showcased practical benefits of this approach in terms of convergence speed, scaling, and applications to black-box AE models on several datasets. Several questions are still open for future work.

In terms of theoretical results, we plan on working on the stability between the diffeomorphic interpolations computed on a dataset and its subsamples. This requires some control over the locations

[5] As we noticed that 3. was a consistent threshold for detecting whether the representative cycle of the most persistent PD point changed between iterations.

of the critical points, which we expect to be possible in *statistical estimation*; indeed sublevel sets of density functions are know to have stable critical points [10, Lemma 17].

Concerning the AE experiment, we plan to investigate the limitations presented in the figure below:  as the initial LSs have zero (left) or two (right) loops, it is impossible to unfold them with diffeomorphisms; instead the optimized latent spaces either also exhibit no topology, or mixes different angles. In future work, we plan on investigating other losses or gradient descent schemes for diffeomorphic topological optimization, including stratified procedures similar to [27] that allow for local topological changes during training.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: a stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

[3] Tolga Birdal, Aaron Lou, Leonidas Guibas, and Umut Simsekli. Intrinsic dimension, persistent homology and generalization in neural networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, volume 34, pages 6776–6789. Curran Associates, Inc., 2021.

[4] Mathieu Carrière, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. Optimizing persistent homology based functions. In *38th International Conference on Machine Learning (ICML 2021)*, volume 139, pages 1294–1303. PMLR, 2021.

[5] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. PersLay: a neural network layer for persistence diagrams and new graph topological signatures. In *23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, pages 2786–2796. PMLR, 2020.

[6] Mathieu Carrière, Steve Y. Oudot, and Maks Ovsjanikov. Stable topological signatures for points on 3d shapes. *Computer Graphics Forum*, 34(5):1–12, 2015.

[7] Frédéric Chazal, David Cohen-Steiner, Leonidas Guibas, Facundo Mémoli, and Steve Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.

[8] Frédéric Chazal, Vin De Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014.

[9] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Subsampling methods for persistent homology. In *32nd International Conference on Machine Learning (ICML 2015)*, volume 37, pages 2143–2151. PMLR, 2015.

[10] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Robust topological inference: distance to a measure and kernel distance. *Journal of Machine Learning Research*, 18(159):1–40, 2018.

[11] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Alessandro Rinaldo, Aarti Singh, and Larry Wasserman. On the bootstrap for persistence diagrams and landscapes. *Modelirovanie i Analiz Informatsionnykh Sistem*, 20(6):111–120, 2013.

[12] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Stochastic convergence of persistence landscapes and silhouettes. *Journal of Computational Geometry*, 6(2):140–161, 2015.

[13] Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in euclidean spaces. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 232–241. ACM, 2008.

[14] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582. PMLR, 2019.

[15] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

[16] James R Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A Zimmer, Julia A Schnabel, and Andrew P King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE transactions on pattern analysis and machine intelligence*, 44(12):8766–8778, 2020.

[17] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[18] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l p-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010.

[19] Meryll Dindin, Yuhei Umeda, and Frederic Chazal. Topological data analysis for arrhythmia detection through modular neural networks. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33*, pages 177–188. Springer, 2020.

[20] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[21] Marcio Gameiro, Yasuaki Hiraoka, and Ippei Obayashi. Continuation of point clouds via persistence diagrams. *Physica D: Nonlinear Phenomena*, 334:118–132, 2016.

[22] Thomas Gebhart and Paul Schrater. Adversary detection in neural networks via persistent homology. *arXiv preprint arXiv:1711.10056*, 2017.

[23] Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *37th International Conference on Machine Learning (ICML 2020)*, volume 119, pages 4314–4323. PMLR, 2020.

[24] Max Horn, Edward de Brouwer, Michael Moor, Bastian Rieck, and Karsten Borgwardt. Topological Graph Neural Networks. In *10th International Conference on Learning Representations (ICLR 2022)*. OpenReviews.net, 2022.

[25] Xiaoling Hu, Li Fuxin, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 5657–5668. Curran Associates, Inc., 2019.

[26] Théo Lacombe, Yuichi Ike, Mathieu Carrière, Frédéric Chazal, Marc Glisse, and Yuhei Umeda. Topological Uncertainty: monitoring trained neural networks through persistence of activation graphs. In *30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pages 2666–2672. International Joint Conferences on Artificial Intelligence Organization, 2021.

[27] Jacob Leygonie, Mathieu Carrière, Théo Lacombe, and Steve Oudot. A gradient sampling algorithm for stratified maps with applications to topological data analysis. *Mathematical Programming*, pages 1–41, 2023.

[28] Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*, pages 1–63, 2021.

[29] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In *37th International Conference on Machine Learning (ICML 2020)*, volume 119, pages 7045–7054. PMLR, 2020.

[30] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). In *Technical Report CUCS-005-96*, 1996.

[31] Arnur Nigmetov, Aditi S Krishnapriyan, Nicole Sanderson, and Dmitriy Morozov. Topological regularization via persistence-sensitive optimization. *arXiv preprint arXiv:2011.05290*, 2020.

[32] Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *Discrete & Computational Geometry*, pages 1–35, 2024.

[33] Steve Y Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Society, 2015.

[34] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pages 13–25. Wiley Online Library, 2018.

[35] Primoz Skraba and Katharine Turner. Wasserstein stability for persistence diagrams. *arXiv preprint arXiv:2006.16824*, 2020.

[36] Yitzchak Solomon, Alexander Wagner, and Paul Bendich. A fast and robust method for global topological functional optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 109–117. PMLR, 2021.

[37] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.6.0 edition, 2022.

[38] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

[39] Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017.

[40] Laurent Younes. *Shapes and diffeomorphisms*. Springer-Verlag, 2010.

A A more extensive presentation of TDA

The starting point of TDA is to extract *quantitative* topological information from structured objects—for example graphs, points sampled on a manifold, time series, etc. Doing so relies on a piece of algebraic machinery called *persistent homology* (PH), which informally detects the presence of underlying topological properties in a *multiscale* way. Here, estimating topological properties should be understood as inferring the number of connected components (topology of dimension 0), the presence of loops (topology of dimension 1), of cavities (dimension 2), and so on in higher dimensional settings.

Simplicial filtrations. Given a finite simplicial complex[6] K, a *filtration* over K is a map $t \mapsto K_t \subseteq K$ that is non-decreasing (for the inclusion). For each $\sigma \in K$, one can record the value $t(\sigma) := \inf\{t \,|\, \sigma \in K_t\}$ at which the simplex σ is inserted in the filtration. From a topological perspective, the insertion of σ has exactly one of two effects: either it creates a new topological feature in K_t (e.g., the insertion of an edge can create a loop, that is, a one-dimensional topological feature) or it destroys an existing feature of lower dimension (e.g., two independent connected components are now connected by the insertion of an edge). Relying on a matrix reduction algorithm [20, §IV.2], PH identifies, for each topological feature appearing in the filtration, the *critical pair* of simplices (σ_b, σ_d) that created and destroyed this feature, and as a byproduct the corresponding *birth* and *death* times $t(\sigma_b), t(\sigma_d)$.[7] The collection of intervals $(t(\sigma_b), t(\sigma_d))$ is a (finite) subset of the open half-plane $\{(b,d) \in \mathbb{R}^2 \,|\, b < d\}$, called the *persistence diagram* (PD) of the filtration $(K_t)_t$. The distance of such a point (t_b, t_d) to the diagonal $\{b = d\}$, namely $2^{-\frac{1}{2}}|t(\sigma_b) - t(\sigma_d)|$, is called the *persistence* of the corresponding topological feature, as an indicator of "for how long" could this feature be detected in the filtration $(K_t)_t$.

Note that if (σ_b, σ_d) is a critical pair for our filtration $(K_t)_t$, it holds that $|\sigma_b| = |\sigma_d|+1$. The quantity $|\sigma_b| - 1$ is the dimension of the corresponding topological feature (e.g., loops, which are created by the insertion of an edge and killed by the insertion of a triangle, are topological features of dimension one). From a computational perspective, deriving the PD of a filtration $(K_t)_t$ is empirically[8] quasi-linear with respect to the number of simplices in K (which can still be extremely high in the case of Vietoris-Rips filtration—see below—where $K = 2^V$ with $|V| = n$ being typically quite large).

The Vietoris-Rips filtration. A particular instance of simplicial filtration that will be extensively used in this work is the Vietoris-Rips (VR) one. Given $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ a point cloud of n points in dimension d, one consider the simplicial complex $K = 2^X$ and then the filtration $(K_t)_t$ defined by

$$\sigma = \{x_{i_1} \ldots x_{i_p}\} \in K_t \iff \forall j, j' \in \{1, \ldots, p\},\ \|x_{i_j} - x_{i_{j'}}\| \leq t. \tag{8}$$

The corresponding persistence diagram will be denoted, for the sake of simplicity, by $\mathrm{Dgm}(X)$.

Note that for $t < 0$, $K_t = \emptyset$, when $t \geq \mathrm{diam}(X)$, $K_t = K$, and there is always a point with coordinates $(0, +\infty)$ in $\mathrm{Dgm}(X)$ accounting for the remaining connected component when $t \to \infty$. This is the unique point in $\mathrm{Dgm}(X)$ for which the second coordinate is $+\infty$ (and is often discarded in practice, as it does not play any significant role). Intuitively, $\mathrm{Dgm}(X)$ reflects the topological properties that can be inferred from the *geometry* of X; this can be formalized by various results which state, roughly, that if the x_i are i.i.d. samples from a regular measure μ supported on a submanifold $\mathcal{M} \subset \mathbb{R}^d$, then with high probability the topological properties of $\mathcal{M}$ are reflected in $\mathrm{Dgm}(X)$ (see [13, 9]). From a computational perspective, note that the VR filtration only depends on X through the pairwise distance matrix $(\|x_i - x_j\|)_{1 \leq i,j \leq n}$, and thus the complexity of computing $\mathrm{Dgm}(X)$ depends only linearly in d[9]. On the other hand, since $\mathrm{Dgm}(X)$ scales (at least) linearly

[6] A simplicial complex is a combinatorial object generalizing graphs and triangulations. Given a finite set of vertices $V = \{v_1, \ldots, v_n\}$, a finite simplicial complex K is a subset of 2^V (whose elements are called *simplices*) such that $\sigma \in K \Rightarrow \tau \in K,\ \forall \tau \subseteq \sigma$ (if a simplex is in the complex, its faces must be in it as well).

[7] It may happen that a topological feature appears at some time t_b and is never destroyed, in which case the death time is set to $+\infty$. However, in the context of the Vietoris-Rips filtration, extensively studied in this work, this (almost) never happens. See the next paragraph.

[8] The theoretical worst case yields a cubic complexity, but the matrix that has to be reduced is typically very sparse, enabling this practical speed up.

[9] However, the statistical efficiency of $\mathrm{Dgm}(X)$ when it is used as an estimator for the topology of an underlying manifold $\mathcal{M}$ deteriorate when the *intrinsic* dimension of $\mathcal{M}$ increases.

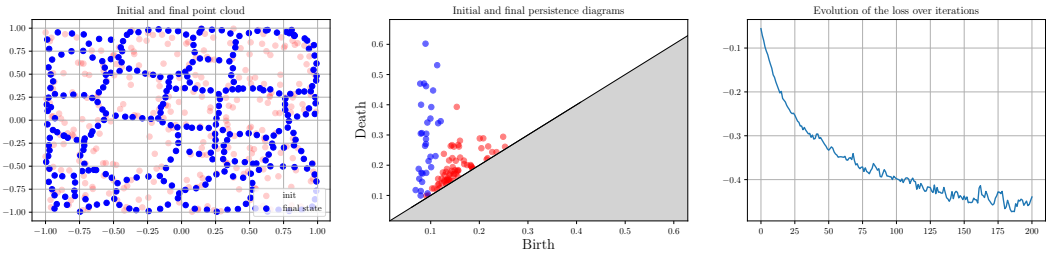


Figure 8: Topological optimization of an initial point cloud X (in red) by minimizing $X \mapsto \sum_{(b,d)\in \mathrm{Dgm}(X)} -|d|^2 + \sum_{x\in X} \mathrm{dist}(x,[-1,1]^2)$. This loss favors the apparition of topological features (loops) while the regularization term penalizes points that would go to infinity otherwise.—Experiment reproduced following the setting of [4], using code available at https://github.com/GUDHI/TDA-tutorial/blob/master/Tuto-GUDHI-optimization.ipynb.

with respect to the number of simplices in K, computing the whole VR diagram of a point cloud $X \in \mathbb{R}^{n\times d}$ can take up to $O(2^n)$ operations. Even if one restricts topological features of dimension $d' \leq d$ (e.g. $d' = 1$ if one only considers loops)—as commonly done—the complexity is of order $O(n^{d'+2})$, which becomes quickly intractable if n is large, even if $d' = 1$ or 2.

B Delayed proofs

Proof of Proposition 3.2. One has: $\|\tilde{v}(x) - \tilde{v}(y)\|_1 = \|\sum_{i\in I}(K(x,x_i) - K(y,x_i))(-\mathbf{K}^{-1}\nabla L(\tilde{X}(t)))_i\|_1 \leq \sum_{i\in I} |\rho_\sigma(\|x-x_i\|) - \rho_\sigma(\|y-x_i\|)| \cdot \|(-\mathbf{K}^{-1}\nabla L(\tilde{X}(t)))_i\|_1$, since we are using Gaussian kernels. As $|\rho_\sigma(\|x-x_i\|) - \rho_\sigma(\|y-x_i\|)| \leq C_{d,\sigma}\|x-y\|_2$, with $C_{d,\sigma} = 2^{\frac{d+1}{2}}\pi^{\frac{d-1}{2}}\sigma^{d-1}$ (see [2, Theorem 8]), it follows that $\|\tilde{v}(x) - \tilde{v}(y)\|_1 \leq C_{d,\sigma}\|x-y\|_2 \cdot \|\mathbf{K}^{-1}\|_1 \cdot \|\nabla L(\tilde{X}(t))\|_1$.

Let us upper bound the term $\|\nabla L(\tilde{X}(t))\|_1$. Let us start with the simplification loss, one has $\frac{\partial \ell}{\partial b_i} = 2(b_i - d_i)$ and, writing $b_i = \|x_{i_1} - x_{i_2}\|_2$ (for some critical points $x_{i_1}, x_{i_2} \in \tilde{X}(t)$), one has $\frac{\partial b_i}{\partial x_{i_1}} = \frac{x_{i_1}-x_{i_2}}{\|x_{i_1}-x_{i_2}\|_2}$ and $\frac{\partial b_i}{\partial x_{i_2}} = -\frac{x_{i_1}-x_{i_2}}{\|x_{i_1}-x_{i_2}\|_2}$. Similarly, one has $\frac{\partial \ell}{\partial d_i} = -2(b_i - d_i)$, and writing $d_i = \|x_{i_3} - x_{i_4}\|_2$ provides the corresponding partial derivatives.

Applying (3), this gives:

$$\|\nabla L(\tilde{X}(t))\|_1 = \sum_{x\in\tilde{X}(t)} \Big\| \sum_{x \overset{b,1}{\to} (b_i,d_i)} 2(b_i-d_i)\frac{x-x_{i_2}}{\|x-x_{i_2}\|_2} - \sum_{x \overset{b,2}{\to} (b_i,d_i)} 2(b_i-d_i)\frac{x_{i_1}-x}{\|x_{i_1}-x\|_2}$$
$$- \sum_{x \overset{d,1}{\to} (b_i,d_i)} 2(b_i-d_i)\frac{x-x_{i_4}}{\|x-x_{i_4}\|_2} + \sum_{x \overset{d,2}{\to} (b_i,d_i)} 2(b_i-d_i)\frac{x_{i_3}-x}{\|x_{i_3}-x\|_2}\Big\|_1,$$

where $\overset{b,1}{\to}$ (resp. $\overset{b,2}{\to}$) means that x appears as left point (resp. right point) in the computation of the birth filtration value b_i of one of the k PD points $(b_i,d_i) \in \mathrm{Dgm}(\tilde{X}(t))$ associated to the loss, and similarly for death filtration values. A brutal majoration finally gives $\|\nabla L(\tilde{X}(t))\|_1 \leq 2\sqrt{d}\sum_{x\in\tilde{X}(t)}\sum_{x\to(b_i,d_i)}|b_i-d_i| \leq 8\sqrt{d}\cdot \mathrm{Pers}_k(\mathrm{Dgm}(\tilde{X}(t)))$, as there are at most four points associated to every $(b_i,d_i) \in \mathrm{Dgm}(\tilde{X}(t))$. One can easily see that the same bound applies to the augmentation loss.

Let us finally bound $\|\mathbf{K}^{-1}\|_1$, one has $\|\mathbf{K}^{-1}\|_1 = \kappa(\mathbf{K})/\|\mathbf{K}\|_1 \leq \kappa(\mathbf{K})$. Indeed, as we are using Gaussian kernels, $\|\mathbf{K}\|_1 = \max_{1\leq i\leq n}\sum_{j=1}^n \rho_\sigma(\|x_i - x_j\|_2) \geq 1$. □

C Complementary experimental results and details

Subsampling and improving over [4]. We reproduce the experiment of [4, §5], see also Figure 8, but starting from an initial point cloud X_0 of size $n = 2000$ instead of $n = 300$. This makes the

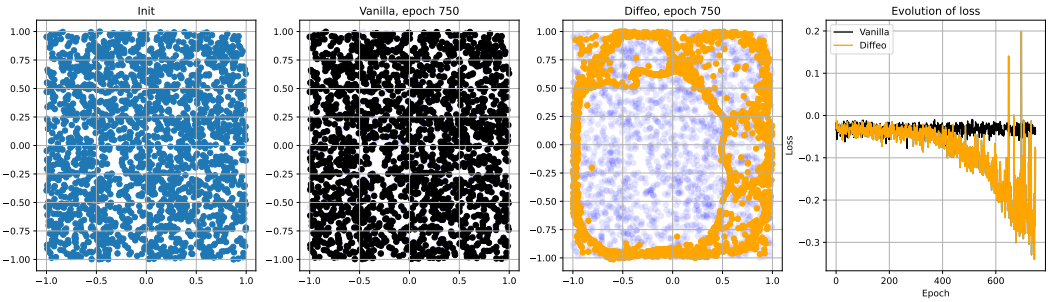


Figure 9: Topological optimization with subsampling. From left to right, the initial point cloud X_0, the point cloud after 750 steps of vanilla gradient descent (+subsampling), the point cloud after 750 steps of diffeomorphic interpolation gradient descent (+subsampling), loss evolution over epochs. Parameters: $\lambda = 0.1$, $\sigma = 0.1$.

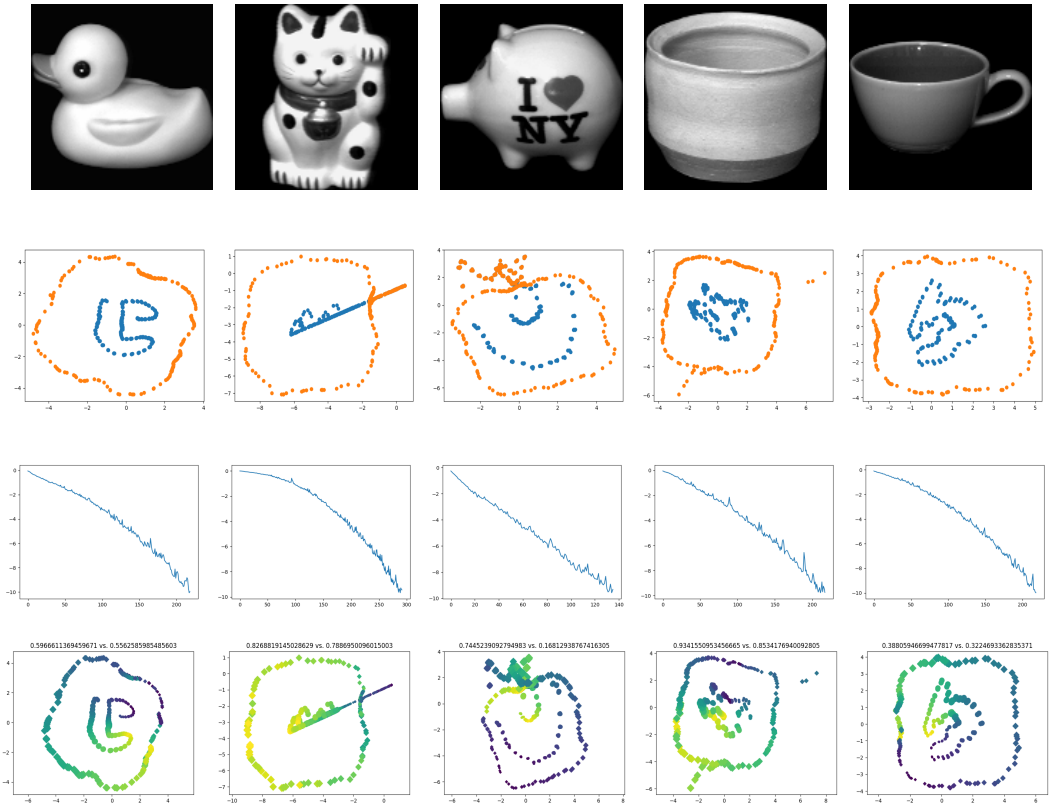


Figure 10: Topologically-optimized LSs for duck, cat, pig, vase and teapot.

raw computation of $\mathrm{Dgm}(X_k)$ *at each gradient* step unpractical. Following Section 3.2 we rely on subsampling with sample size $s = 100$ and apply Algorithm 1. Results are summarized in Figure 9. While relying on vanilla gradients and subsampling barely changes the point cloud even after 750 epochs, the diffeomorphic interpolation gradient with subsampling manages to decrease the loss.

More extensive reports of running time and comments. On the hardware used in our experiments (The first two experiments were run on a `11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz`, the last one on a `2x Xeon SP Gold 5115 @ 2.40GHz`.), we report the approximate following running times:

- Small point cloud optimization without subsampling (see Figure 4, $n = 200$ points): one gradient descent iteration takes about 1s for vanilla and Diffeo (our). The use of `oineus` integrated in our pipeline raises the running time (per iteration) to 10 to 20 seconds. Note

that Diffeo and `oineus` may converge in less steps than Vanilla, preserving a competitive advantage. We also believe that `oineus` has a significant room for improvement in terms of running times and may be a promising method in the future to be used jointly with our diffeomorphic interpolation approach.

- Iterating over the stanford bunny with subsampling ($n = 35,947$, $s = 100$) takes about 3 seconds per iteration for Vanilla and 20 second for our diffeomorphic interpolation method. The increase in running time with respect to the previous experiment mostly lies on instantiating and applying the $n \times d$ ($d = 3$) vector field $\tilde{v}$ (requires to compute $\rho_i(x - x_i)$ for each new x (n of them) and sampled x_i ($|I|$ of them, which is typically very small), hence a $\sim O(n)$ complexity).
- Training the VAE for the `COIL` dataset is the most computationally expensive part of this work: it takes about 3 hours per shape (20 of them). In contrast, performing the topological optimization take few dozen of minutes (less than one hour) for each shape. Applying it is done in few seconds at most. Recall that our method is designed to handle pre-trained models (which may be way more sophisticated than the one we used!); and its running time does not depend on the complexity of the model.